Smart Camp

Building Scalable and Highly Available IT-Infrastructures

Sergej Proskurin
Furtwangen University, Germany

David McMeekin
Curtin University, Australia

Achim P. Karduck
Furtwangen University, Germany
Curtin University, Australia

Abstract — **The Western Australian resources boom has created a demand for a large amount of domestic accommodations, known as mining camps. However, due to the absent infrastructure within the remote regions of Australia, the energy supply of these mining camps is expensive. In order to reduce the electricity consumption of the mining camps, the Smart Camp project was initiated. The system infrastructure consists of a home automation based controller, placed in each mining accommodation unit to reduce energy consumption, and a centralized management unit, coordinating the controllers. Due to the fact that the size and complexity of mining camps may grow over time, the provided infrastructure of the management unit has to be able to evolve. One possible solution is to design a system in the context of high availability and horizontal scalability. This paper proposes a horizontally scalable and high availability infrastructural concept, in the context of the Smart Camp project. This concept also utilizes cost effective open source solutions running on commodity hardware. Within the context of horizontal scalability and reliability, this paper provides an applied research outline of some of the real world considerations, such as open source based high availability, load balancing, and distributed database solutions.**

Keywords: Smart Camp, Scalability, High Availability, Load Balancing, Distributed Database, Infrastructure.

I. Introduction

The purpose of the Smart Camp project was the need for a reduction of the electricity costs within mining camps in Australia. To resolve high energy costs issues within mining camps, the Smart Camp project comprises the development of an intelligent controller, here referred to as the Smart Home Controller (SHC) and a management unit, here referred to as the Smart Camp Management Unit (SCMU). Within the scope of the Smart Camp project, the SHC's will be installed within each mining accommodation unit of the camp and hence aiming to transform it into a Smart Home. In order to reduce the overall energy costs, the SHC will observe and control certain electrical devices within Smart Home's. Since air conditioners represent the main electricity consumers within mining camps, the particular controllers, first of all, will be able to control the air conditioner and with it the main electricity consumption in each Smart Home. Each SHC will act considering residents presence, the working hours, and further preferences. To obtain the required preferences, the controller will send specific requests to the SCMU, which will contain all required information for every controller within the mining camp. The management unit will be central in each mining camp, in order to monitor the state of each controller and to provide them with specific services (Figure 1).

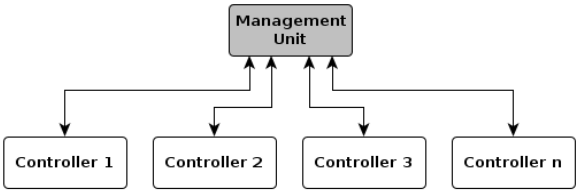

Figure 1: The Smart Camp System

Within the context of the Smart Camp project, a prototype has already been developed and deployed within the mining camp in Karratha, Western Australia. The initially proposed prototype infrastructure revealed multiple weaknesses; It provided single points of failure and was not capable to keep up with the growing number of SHC requests. According to mining camp setup, the final Smart Camp system will consist of about 1.500 SHC's, which have to be coordinated by the central SCMU within the camp. Additionally, the particular mining camp may be extended after time, in terms of adding more Smart Homes's and Smart Home Controllers to the camp, thus significantly increasing network demands. Therefore, the management unit will have to be able to process all incoming requests and provide required information to all controllers within the mining camp, independent of the number of Smart Homes. In case of a mining camp expansion, the management unit must not cause a bottleneck and it has to be able to cope with the increased number of requests. Consequently, as individual mining camps may grow over time both the Smart Camp Management Unit and its infrastructure must be designed in such a way to be able to be easily scalable.

Above design requirement is in line with today's key requirement for system architectures and infrastructures, that they have to be designed with foresight in terms of scalability, availability, and evolvability. Provided services within infrastructures must remain available, even in case of one or more hardware and/or software outages. Single points of failure may lead to serious and in particular expensive consequences, such as breakdown of all available services. Moreover, infrastructures must avoid potential bottlenecks by being easily expandable, considering costs of required components. For this reason, this paper deals with the reasonable question: "how to build cost effective scalable and highly available IT infrastructures?". In brief, highly available

systems eliminate single points of failure by implementing redundant components. Scalable systems enable easy adaption of the particular system to the growing demands. Our research implies the analysis and familiarization of scalable and reliable mechanisms based on open source solutions and their adoption within the Smart Camp context. In the end, the aim is to develop a highly scalable and highly available infrastructural concept for the SCMU, which will be able to be adapted easily and without expensive hardware to the growing services and network requirements of a mining camp, or another Smart Home based infrastructure.

II. Related Work

Most systems are developed to meet only the current requirements of a particular problem. However, requirements on systems change regularly over time. Within the Smart Camp context, requirements on mining camps, such as the number of Smart Home Controller requests, may grow over time. As consumer demands grow, the need for more system resources increases. Obviously, if the demand shrinks, using too many resources in order to solve the need would be a waste of money. In both cases the system will either not work properly or waste a lot of system resources. In order to adapt the system to the changed requirements, the system has to be either extended or reduced. Adaption of a non-scalable system to changed requirements often leads to complete infrastructural redesign, inducing high costs, and thus operational loss. Additionally, a complete infrastructural redesign cannot be performed without significant time investments. On the other hand, a scalable system ideally can be adapted cost effectively to the changed requirements within a short period of time and without the need for a complete infrastructural redesign.

According to Theo Schlossnagle, scalability means “how well does a particular solution fit a problem as the scope of that problem increases”[1] or decreases. In this context, to scale up or to scale out a system means to extend the systems resources. In contrast, to scale down a system means to reduce the systems resources. In general, scalability is about the adaption of a system to variable needs of a problem, by increasing or reducing its system resource capacity. This is required e.g. in order to provide services to a much higher number of clients. A distinction is made between vertical and horizontal scalability:

- A system is vertically scalable if the capacity of this system can be increased by replacing internal hardware components with more high performance components, such as RAM, processors, storage, etc.[1]
- A system is horizontally scalable if the resource capacity of this particular system can be increased by adding additional hardware and/or software components of the same type[1].

While a vertically scalable system is both very expensive and limited in its performance enhancement, which can be led back to Moore's Law[2], a horizontally scalable system is often a much more cost effective solution, performing better as its counterpart. On the other hand, additional components in horizontal scalable systems create a distributed system, which is not easy to deal with. Nevertheless, the vertical scalability approach should be considered only if solving problems using the horizontal scalability approach is not an option, as the achieved performance enhancement is both too limited and too expensive[1].

In order to develop scalable and highly available systems, a combination of the following methodologies must be considered:

- High availability mechanisms
- Load balancing mechanisms
- Database distribution mechanisms
- Caching mechanisms

Although the individual methodologies do not present new solutions, their combination opens doors to innovative highly scalable system possibilities, required by a growing number of companies. A detailed introduction of the used approaches and their combination for Smart Camp follows.

In his book, Theo Schlossnagle describes a variety of potential possibilities of how to create scalable systems[1]. He conveys a fundamental understanding and possible configurations of the above mentioned mechanisms including their advantages and difficulties. Additionally, Willy Tareau covers load balancing fundamentals, representing one of the central elements of scalable systems[6]. In order to understand advantages, difficulties, and limits of database distribution, it is essential to internalize the concepts of Eric Brewer's CAP Theorem[4]. Additionally, Christof Strauch presents in his PhD Thesis a very detailed overview of highly scalable Non-Relational Database Management Systems (NoSQL)[3], which can be compared to distributed Relational Database Management Systems (RDBMS) in Eben Hewitt's book[5]. Refer to [11] for a detailed outline of the results presented in this paper.

III. System Architecture

This section consists of three subsections; The scope of the first subsection (subsection A.) introduces the limitations of the initial Smart Camp infrastructure. The second subsection (subsection B.) introduces the chosen components for a scalable and highly available infrastructure, which best fit the Smart Camp requirements. The third and last subsection (subsection C.) of this section presents the developed concept of a scalable and highly available infrastructure within the Smart Camp context, which can be easily adapted to requirements of other institutions.

A. Initial System Architecture

The initial Smart Camp system implementation consisted of only two layers: The Smart Home Controllers and the actual Smart Camp Management Unit as shown in Figure 1 [12]. The SHC’s are responsible to observe and to actively control the state of the associated accommodation units, whereas the actual state of the accommodation units is associated in this context with the state of multiple devices within the Smart Homes, such as air conditioner, television, and multiple sensors attached to the particular SHC. By this way, every SHC is responsible for autonomous observation and control of the associated Smart Homes. The corresponding SCMU is responsible to manage all SHC’s within the Smart Camp and to provide them with required information. The SCMU within the

initial Smart Camp system consists of one web server and one database server configured on only one physical machine. The SCMU contains all configuration information for each registered SHC within the mining camp.

However, the initial system architecture comprises an SHC within every Smart Home of the mining camp. A block of Smart Homes or a Smart Home Complex implies four Smart Homes. The actual Smart Camp may consist of any number of such Smart Home Complexes (Figure 2).

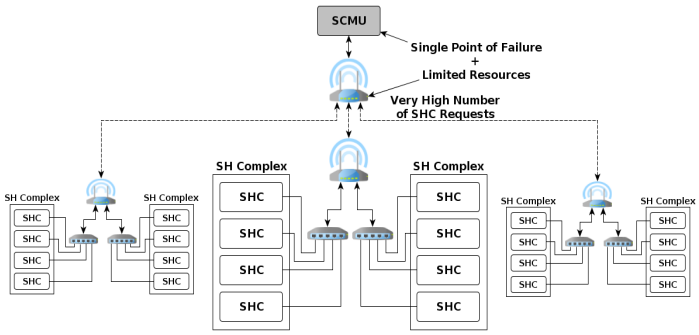

Figure 2: Initial Architecture

In order to establish a network communication between the SHC's in every Smart Home and the SCMU, the following network infrastructure is used: Each SHC within a Smart Home Complex is connected to a Layer 2 switch. Every two Smart Home Complexes are connected to a Layer 3 switch, which establishes a Wireless LAN (WLAN) communication with another layer 3 switch, connected to the SCMU (Figure 2).

Since the management unit consists of only one physical machine, its resources are limited to the performance of only this particular machine. If the number of Smart Homes within the mining camp grows, at some point the management unit will slow down dramatically or even will not be able to process all incoming requests any more. Furthermore, in the event of only one hardware and/or software failure on the management unit itself or on the corresponding layer 3 switch, the controller's will not be able to access the provided services of the management unit any more, thus making the system not reliable at all. This means, that the infrastructure of the initial Smart Camp system is not scalable and its provided services are not highly available. As long as the infrastructure within a mining camp is not scalable, it will not be possible to extend a mining camp, without a complete infrastructural redesign. As long as the provided services are not highly available, crashes of single components of the management unit may lead to outages of all provided services, increasing maintenance costs. For these reasons, this paper elaborates a new, scalable, and highly available infrastructural concept for the management unit, which allows to extend a mining camp to almost any number of Smart Homes. In order to build a scalable and highly available architecture, the following building blocks will be motivated and described.

B. Components

This paper proposes a concept of a horizontal scalable and highly available infrastructure within the context of large scale systems. This concept utilizes cost effective open source solutions running on commodity hardware implying high availability, load balancing, and distributed database components.

1) High Availability Component

In general, an IT infrastructure, no matter how highly scalable it is, is highly vulnerable, as long as it provides single points of failure. Breakdowns of single hardware and/or software components may lead to a complete halt of the system. To avoid complete breakdowns of the system, all system components must be designed redundantly. High availability approaches try to eliminate single points of failure by means of adding redundant hardware and/or software components to a consistent system. This approach increases reliability of the overall system by making it fault tolerant. "A single point of failure exists when a critical function is provided by a single component. If that component fails, the system has no other way to provide that function and essential services become unavailable. The key facet of a highly available system is its ability to detect and respond to changes that could impair essential services"[7].

There exist two basic approaches how to make a system highly available: Depending on the functionality of the system, the components can either be set up in a master slave or in a multi master cluster configuration. The first approach introduces a master slave or an active-passive configuration; It requires at least two equally configured physical components, in order to provide high availability of specific services. One of the components will be elected as master and the remaining as slaves, thus creating a highly available cluster. Cluster members monitor each other by sending periodical messages or heart beats to other cluster components, in order to present their availability. If one component does not present its availability, the cluster members will wait a certain period of time until the particular component will be considered as down or not available and subsequently removed from the cluster. In case the master becomes unavailable, one of the slave components will take over the provided service responsibilities. In a master slave configuration, the slaves serve only to take over the provided services in the event of a master crash. Because of the fact that at least one passive component waits the most of the time to take over and does not provide any services, the system throughput is limited to the performance of only one active component. The second approach presents a multi master or an active-active configuration with at least two masters. This solution differs to a master slave configuration in such a way that it provides multiple active components at the same time, thus highly increasing the system throughput. In the Smart Camp scenario, a multi master High Availability configuration will be preferred.

Wackamole is an open source based high availability solution, originally developed at the John Hopkins University's Center for Networking and Distributed Systems[8]. Wackamole runs on top of the group communication toolkit Spread, where every participant presents a member of a particular group. Spread is able to detect and to handle group changes, implying the detection of both new group members and node failures. Additionally, Spread enables a dynamic recovery from network partitions. In comparison to other high availability solutions, such as Heartbeat, Wackamole provides the possibility to create peer-to-peer based high available clusters in either active-passive or active-active configurations,

including a high number of cluster components, as preferred by the project. Therefore, it provides a configurable number of virtual IP addresses (VIP) representing cluster entry points, which will be equally distributed across all cluster components. In general, a VIP is a logical IP address, which is not fixed to a specific physical network interface, making it able to share the address between different components. Each cluster component may be responsible for multiple VIPs, whereas a certain VIP may belong at the same time to maximum one cluster component. In case of node failures, the cluster rebalances the VIPs across the available nodes. That implies, that as long as at least one cluster component is up and running, all published VIPs remain available. A VIP failover is completely transparent for the clients (Figure 3).

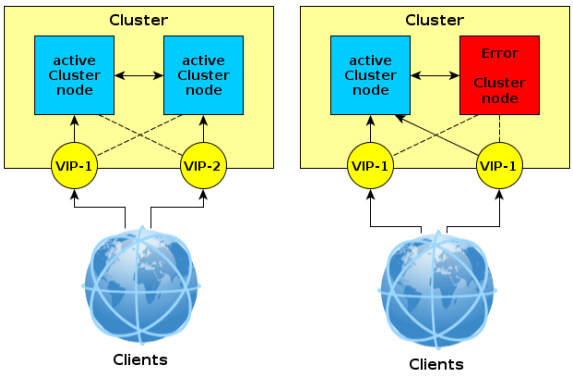

Figure 3: Wackamole Active-Active Configuration

2) *Load Balancing Component*

In the context of horizontal scalability, systems are able to be expanded with help of additional components, called load balancers. This, to be able to increase their data capacity, processing power, and to handle a growing number of client requests. Therefore, the growing workload has to be spread across multiple physical machines responsible for the same task, such as web servers. The general idea behind load distribution is to equally utilize multiple physical machines in order to combine their resources to handle higher loads. In horizontal scalable systems, load balancers present main components, as they are able to distribute incoming loads horizontally across multiple equally configured machines. Load balancers unify multiple machines into clusters, reducing the utilization of the individual components and simultaneously increasing the overall throughput. In highly scalable systems, load balancers may be installed e.g. in front of firewall-, Web-, and cache servers; This allows to independently scale out each infrastructural layer.

Linux Virtual Server (LVS) is a Linux Kernel patch, which enables high performance load balancing on OSI layer 4[9]. In comparison to other software based load balancing solutions on higher OSI layers, LVS is able to handle multiple 100.000 of simultaneous client requests. As the present SCMU infrastructure does not require any features provided by higher OSI layers slowing down the overall provision of client requests, LVS was selected for the project. LVS distributes incoming load across multiple components, thus creating a cluster. All cluster functionality, such as addition of new components, takes place transparently for the clients. LVS supports three different architectural approaches, namely "Network Address Translation" (NAT), "Direct Routing", and "IP Tunneling". IP Tunneling enables, in comparison to NAT and Direct Routing configurations, a load distribution across multiple machines distributed across different networks. In comparison to other open source based solutions, such as HAProxy or Apache's ModBackhand, LVS is not able to provide health checks without additional software solutions, such as LDirectord; LDirectord was specially designed for monitoring and administration of physical servers within a LVS cluster (Figure 4).

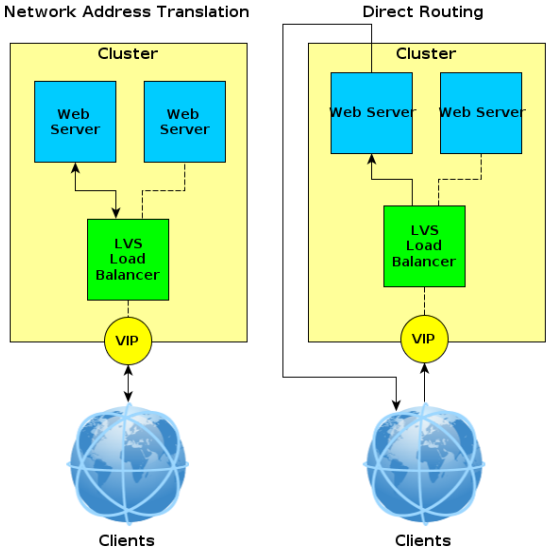

Figure 4: Linux Virtual Server

3) *Distributed Database Component*

Today's requirements are growing and so does the need for storing more data in a faster way, as well. Furthermore, the stored data should be highly available. Scaling up databases vertically means buying more, bigger, and faster hard drives, RAM, and processors and connecting them to the database to satisfy the needs for the moment: Vertical scalability is both, as outlined earlier, limited and very expensive. But even if there were no limitations of vertical scalability and no need to pay attention to costs, what would be a solution for high availability? Even the biggest database server is not able to provide high availability as long as it represents a single point of failure. Besides, if it comes to horizontal scalability of a consistent IT-infrastructure, databases often represent a bottleneck of the system, as only one machine is able to handle a limited number of both read and write operations. In summary, the more operations are to be performed, the slower the database becomes. In order to handle a growing number of read and write operations, thus satisfying the growing requirements of a horizontally scalable infrastructure, a database must be able to scale out horizontally and hence distribute the data across multiple physical machines, thus increasing both the data capacity and performance of the database.

In his theorem, Eric Brewer states three characteristics which have to be considered when designing a distributed

database system: Consistency, Availability, and Partition-Tolerance[4]. According to Brewer, a distributed database system can meet only two of these characteristics at any point in time. First, a system is "Consistent" if every system component can see the same data at any time. Second, a system including its data remains "Available", even if one or more hardware and/or software components crash. Third, a system is "Partition Tolerant" if it is able to proceed operations after network failures between components, thus creating several partitions. Additionally, "Partition Tolerance" enables the possibility to dynamically add or remove new system components. Consequently, if a system does not require "Partition Tolerance", such as local distributed systems, it should be able to provide consistency and availability of data. Systems, distributed for example across multiple geographical areas, should be tolerant to network partitions, thus prohibiting availability and consistency at the same time.

Cassandra is a highly scalable NoSQL database, which was originally developed by Facebook[10]. Facebook open sourced Cassandra in 2008. At this time, the project was taken over by Apache and it is now one of its leading projects. In comparison to other NoSQL database solutions, such as MongoDB, Cassandra provides a decentralized peer-to-peer based cluster without centralized management nodes. Its multi master architecture partitions the data to multiple nodes, replicates the individual node content to multiple nodes, and hence eliminates any single points of failure. Cassandra runs on top of Gossip protocol, which detects and handles cluster component and network failures. According to the CAP Theorem, Cassandra provides Availability and Partition Tolerance. That is, Cassandra forfeits strict consistency, but because of its property, called "tuneable consistency", it is possible to increase consistency of both read and write operations to a higher degree.

C. *New System Architecture*

The following infrastructural model has been devised for the management unit of the Smart Camp system, presenting a constellation of the presented open source components of the previous subsection III.B. (Figure 5).

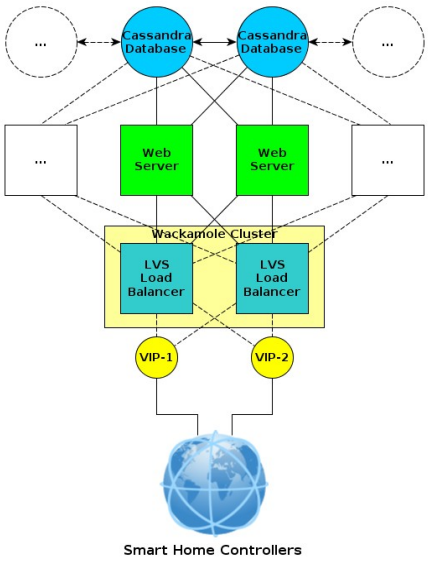

Figure 5: New SCMU Infrastructure

The presented infrastructural solution is both highly available and scalable. Each of the presented layers can be scaled out horizontally independent of other layers. Furthermore, the scalability of each layer does not affect other layers within the infrastructure. Moreover, the individual levels are highly available and can even be considered as N-1 fault tolerant. N-1 fault tolerance means that every component but one of each layer may fail, without affecting the availability of provided services. However, the model implements only three of four presented requirements for scalable infrastructures, namely high availability, load balancing and distributed database solutions. The reason why the model does not provide a caching solution is that the current implementation of the Smart Home Controllers does not require a high number of read operations, which have to be cached. Since the presented infrastructure was designed to be able to be extended, a caching solution may be easily included into the infrastructure, as the requirements of the Smart Camp grow. Furthermore, the individual layers of the infrastructure are not limited to local area network limitations. That is, each of these layers may be placed to different geographical locations.

IV. Assessment

The proposed infrastructure has been simulated within a virtual environment, using six virtual machines. Particular parts of the infrastructure were turned off, in order to inspect its reliability (Figure 6).

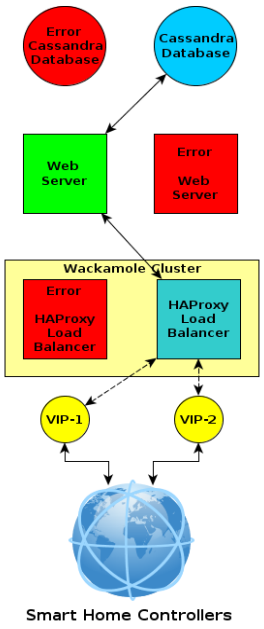

Figure 6: Crash Simulation

It turned out that the proposed infrastructure does not provide any single points of failure, as all services were available after component outages within each logical layer. Additionally, the result showed that every logical layer except for the database layer within the infrastructure is N-1 fault tolerant. As long as at least one component of the affected layers remains available, all services can be accessed without any influence of the SHC activity within the Smart Camp. In this scenario, even the database layer is also N-1 fault tolerant, as it consists of only two components storing the same data.

But as the database cluster grows, the database layer will not remain N-1 fault tolerant any more, as the data size will exceed the data capacity of the remaining database node. In summary, the result demonstrates a very reliable infrastructure for the SCMU within the context of the Smart Camp project. The infrastructure does not provide any single points of failure and even a crash of every component but one within each layer except for the database layer does not affect service and data availability.

In order to increase the overall system resource capacity, it is possible to scale out or to extend the individual layers. Obviously, the complexity of the extension of independent components, such as web servers or load balancers, is much lower than the complexity of the extension of dependent components, such as database nodes. This is also the reason why the scalability of database nodes is limited to a higher degree than independent infrastructural components. However, databases are in general one of the core components within infrastructures and often present bottlenecks, since database operations are very expensive. Tests were performed on a Cassandra cluster consisting of one, two, three, and four cluster nodes, in order to demonstrate the increasing performance and scalability of both read and write operations after addition of more nodes to the cluster, as shown in Figure 7.

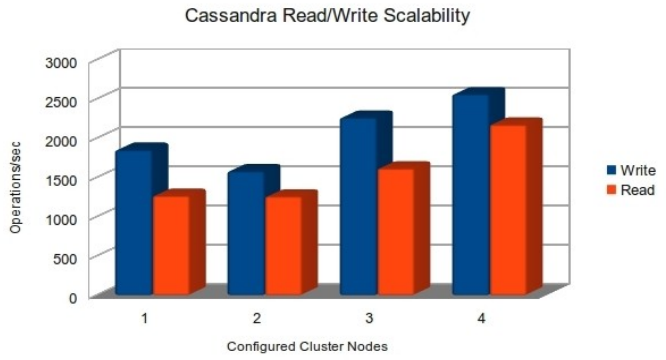

Figure 7: Cassandra Cluster - Operations per Second

The results show that a migration from two cluster nodes to four cluster nodes increases the write operation performance by almost 39% and the read operation performance by almost 43%. The migration from one to two cluster nodes degrades the write operation performance by almost 19% and the read operation performance by not even 2%. On the other hand, two database nodes provide highly available data, thus making the database more reliable and the potential performance degradation less important.

In summary, the results present the scalability of the database solution and a clear performance enlargement of read and write requests. Since all database cluster nodes and the test node were running within multiple virtual machines on only one physical machine and hence limiting the overall performance of the cluster, the results may have inaccuracies. Still, apart from that, it is clear that the addition of database nodes increases the performance and allows to perform a higher number of read and write operations, thus enabling to perform a higher number of simultaneous database requests.

V. Conclusion and Outlook

The developed concept enables the adaption of the Smart Camp Management Unit to the growing requirements of a mining camp. Multiple open source components were selected and evaluated within the scope of the Smart Camp project's requirements. Subsequently, the selected open source components, as depicted in III.B.1), 2), and 3), were integrated into a system in such a way, that the resulting infrastructure became both scalable and highly available. In order to observe scalability and high availability implications of the system, multiple tests within virtual environments were successfully completed and presented within the scope of this paper.

The presented system architecture and infrastructure concept provides the possibility to unify the management of multiple mining camps within one centralized management unit. This means, it is possible to combine multiple mining camps, which will be managed by only one centralized Smart Camp Management Unit.

As the proposed infrastructure was designed to be easily expandable, it is conceivable to apply the presented concepts of a Smart Camp to a Smart City. The Smart City introduces a concept for a scalable and highly available infrastructure, which may be extended with semantic technologies and applied by any company or organization.

Acknowledgments

Thank you very much Markus Lanthaler, as you were very supportive and had always time for a detailed discussion. Many thanks to Professor Elizabeth Chang for providing with DEBII all the support of an inspiring environment.

References

[1] Theo Schlossnagle, "Scalable Internet Architectures", Sams Publishing, July 21, 2006.

[2] Excerpts from a conversation with Gordon Moore: "Moore's Law", ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf.

[3] Christof Strauch, "NoSQL Databases", PhD Thesis, Hochschule der Medien, Stuttgart.

[4] Julian Browne, "Brewer's CAP Theorem", http://www.julianbrowne.com/article/viewer/brewers-cap-theorem, January 11, 2009.

[5] Eben Hewitt, "Cassandra: The Definitive Guide", O'Reilly, 2011.

[6] Willy Tarreau, "Making Applications Scalable with Load Balancing", http://www.exceliance.fr/sites/default/files/biblio/art-2006-making_applications_scalable_with_lb.pdf, September, 2006.

[7] Barbara Lancaster Wedge Greene, "Carrier-Grade: Five Nines, The Myth and The Reality", http://www.ltcinternational.com/inside-out/uploads/ltc_carriergrade_whitepaper.pdf, March 18, 2007.

[8] Wackamole Homepage, http://www.backhand.org/wackamole.

[9] Linux Virtual Server Homepage, http://www.linuxvirtualserver.org.

[10] Cassandra Homepage, http://www.datastax.com/docs/1.0/index.

[11] Sergej Proskurin, "Smart Camp - Scalability in IT Infrastructures", Furtwangen University, 2012.

[12] L. Oslislo, A. Talevski, A.P. Karduck, "Smart Camp: Benefits of Media and Smart Service Convergence", 25th IEEE International Converence on Advanced Information Networking and Applications (AINA 2011), March 22-25, 2011, Singapore.